Diffusion Models

An Introduction and Deep Dive

August 8, 2025

Outline

What are Diffusion Models?

- Diffusion models are inspired by non-equilibrium thermodynamics. They define a Markov chain of diffusion steps to slowly add random noise to data and then learn to reverse the diffusion process to construct desired data samples from the noise.
- Once we are able to achieve that, we can sample a point from the conditional Gaussian distribution corresponding to a particular label and backpropagate to get to a required image from Gaussian noise.

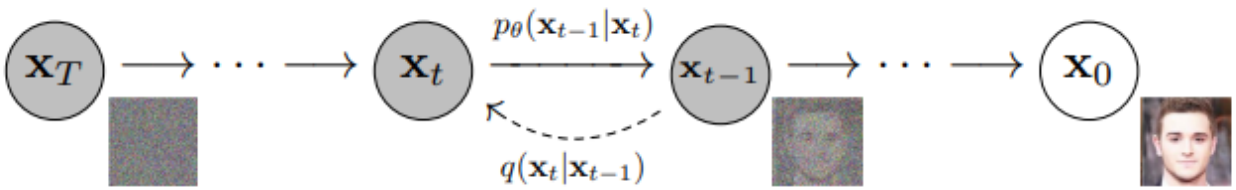


Figure: Directed Graphical representation of Diffusion Model

Forward Diffusion Process

- Given a data point sampled from a real data distribution $x_0 \sim q(x)$, let us define a forward diffusion process in which we add a small amount of Gaussian noise to the sample in T steps, producing a sequence of noisy samples $x_1, x_2, \ldots, x_T$, where step sizes are controlled by a variance schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$.

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I), \quad q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1})$$

- As t becomes larger, x_0 keeps on losing its properties and as $T \to \infty$, x_T converges to an isotropic Gaussian distribution.

x_t in terms of x_0

- Using a reparameterization trick, let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$:

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1} \quad \text{where } \epsilon_{t-1}, \epsilon_{t-2}, \cdots \sim \mathcal{N}(0, \mathbf{I}) \\
&= \cdots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \\
q(\mathbf{x}_t \mid \mathbf{x}_0) &= \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})
\end{aligned}$$

- Usually, we can afford a larger update step when the sample gets noisier, so $\beta_1 < \beta_2 < \cdots < \beta_T$ and therefore $\bar{\alpha}_1 > \cdots > \bar{\alpha}_T$.

Reverse Diffusion Process

- To reconstruct the original image from Gaussian noise, we try to learn the distribution $q(x_{t-1} \mid x_t)$.
- If β_t is small enough, this reverse process is approximately Gaussian.
- Although we cannot compute this distribution exactly, we can derive it conditioned on x_0:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I})$$

Finding $\tilde{\mu}_t$ and $\tilde{\beta}_t$

- Using Bayes' rule:

$$q(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{x}_0) = q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) \frac{q(\mathbf{x}_{t-1} \mid \mathbf{x}_0)}{q(\mathbf{x}_t \mid \mathbf{x}_0)}$$

- Substituting the known Gaussians:

$$\tilde{\beta}_t = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t} \beta_t$$

$$\tilde{\mu}_t = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_t \right)$$

Reverse Diffusion Process

- We'll now learn the backward process via a neural network:

$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

- Objective:

$$\min_\theta -\mathbb{E}_{x_0 \sim q}[\log p_\theta(x_0)]$$

- Using KL divergence:

$$\mathcal{L}_{\text{VLB}} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right] \geq -\mathbb{E}_{q(\mathbf{x}_0)} \log p_\theta(\mathbf{x}_0) \tag{1}$$

Further Manipulation

- After some algebra, the VLB loss decomposes as:

$$\mathcal{L}_{\text{VLB}} = L_T + L_{T-1} + \cdots + L_0 \tag{2}$$

$$\text{where } L_T = D_{\text{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_T)) \tag{3}$$

$$L_t = D_{\text{KL}}(q(\mathbf{x}_t|\mathbf{x}_{t+1}, \mathbf{x}_0) \,\|\, p_\theta(\mathbf{x}_t|\mathbf{x}_{t+1})) \quad \text{for } 1 \leq t \leq T-1 \tag{4}$$

$$L_0 = -\log p_\theta(\mathbf{x}_0|\mathbf{x}_1) \tag{5}$$

REMAINING DIFFUSION SLIDES HERE

Speech Restoration

- Diffusion model discussed above will be able to generate images/speech but cann't be used directly for speech restoration.
- bto deal with speech restoration using diffusion model we need to study something called SDE(Stochastic differential equation.
- The basic idea will be to generate noisy version to speech from clean speech by adding some noise in a directed manner and if we are able to simulate the backward direction of this process somehow(as done in diffusion model) then we will be able to generate clean speech from noisy speech following that reverse process.

What are Stochastic Differential Equations (SDEs)?

- SDEs are differential equations where one or more terms are stochastic processes, meaning they incorporate random noise.
- They are used to model systems that evolve randomly over time.
- Unlike ordinary differential equations (ODEs) which describe deterministic change, SDEs account for inherent uncertainty.

example_random_walk.png

Review: Ordinary Differential Equations (ODEs)

- An ODE describes the rate of change of a variable with respect to another, typically time.
- General form: $\frac{X_t}{t} = f(X_t, t)$
- Solution X_t is a deterministic path given initial conditions.

Example: Simple Growth Model

$$\frac{X_t}{t} = rX_t \quad \Longrightarrow \quad X_t = X_0 e^{rt}$$

- Here, r is the constant growth rate.
- The future value X_t is perfectly predictable.

Review: Stochastic Differential Equations (SDEs)

- An SDE models the rate of change of a system with both deterministic and random components.
- General form (Itô): $X_t = \mu(X_t, t)t + \sigma(X_t, t)W_t$
- Solution X_t is a stochastic process — outcomes are random, even with the same initial conditions.

Example: Geometric Brownian Motion (GBM)

$$X_t = rX_t\, t + \sigma X_t\, W_t \quad \Rightarrow \quad X_t = X_0 \exp\left(\left(r - \frac{1}{2}\sigma^2\right) t + \sigma W_t\right)$$

- r: drift (expected growth rate), σ: volatility.
- Future value X_t is uncertain due to Brownian motion W_t.

Algebraic Equation → ODE → SDE

Algebraic Equation (Deterministic Motion):

$$X_t = X_0 + vt$$

- Object moves at constant velocity v.

Ordinary Differential Equation (ODE):

$$\frac{X_t}{t} = v$$

- Expresses rate of change explicitly.

Stochastic Differential Equation (SDE):

- Suppose the velocity has random fluctuations (e.g., due to noise or uncertainty).
- A naive model: $\frac{X_t}{t} = v + \eta_t$, where η_t is white noise.
- But this is ill-defined in classical calculus.
- Using Itô calculus, we write:

$$X_t = vt + \sigma W_t$$

Key Concept: The Wiener Process (Brownian Motion)

- Also known as Brownian motion, denoted W_t.
- It's the primary source of randomness in SDEs.
- **Properties:**
  1. $W_0 = 0$ (starts at zero).
  2. Independent increments: For $0 \leq s < t < u < v$, $W_t - W_s$ and $W_v - W_u$ are independent.
  3. Stationary increments: $W_t - W_s \sim \mathcal{N}(0, t - s)$.
  4. Continuous paths (almost surely).
  5. Nowhere differentiable (a key distinction from ODEs).

General Form of an Itô SDE

- Most commonly, SDEs are defined in the sense of Itô calculus due to the non-differentiability of the Wiener process.
- An Itô SDE for a process X_t is written as:

$$X_t = f(X_t, t)\, t + g(X_t, t)\, W_t$$

- Where:
  - X_t: **Random Process** (evolve overtime)
  - $f(X_t, t)$: **Drift coefficient** (deterministic part).
  - $g(X_t, t)$: **Diffusion coefficient** (stochastic part).
  - W_t: Infinitesimal increment of a Wiener process.

Ornstein-Uhlenbeck Process

- A classic example of a **mean-reverting stochastic process**.

SDE Form:

$$X_t = \kappa(\theta - X_t)t + \sigma W_t$$

- $\kappa > 0$: **rate of mean reversion**.
- θ: **long-term mean** (the level to which the process tends to return).
- σ: **volatility** of the process.
- W_t: standard Brownian motion.

Properties:

- Has an explicit solution:

$$X_t = \theta + (X_0 - \theta)e^{-\kappa t} + \sigma \int_0^t e^{-\kappa(t-s)} W_s$$

- As $t \to \infty$, $X_t \to \theta$ in distribution.

Mean-Reverting Process

Definition

A stochastic process is said to be **mean-reverting** if it tends to drift toward a long-term average (mean) value over time, even after random fluctuations.

Example: Ornstein-Uhlenbeck Process

$$X_t = \kappa(\theta - X_t)t + \sigma W_t$$

Key Insight: If $X_t > \theta$, drift becomes negative, pulling it down. If $X_t < \theta$, drift becomes positive, pulling it up.

Used In: Modeling interest rates, volatility, neuron potentials, physical forces, etc.

Variance Exploding vs Variance Preserving SDEs

1. Variance Preserving SDE (VP-SDE)

$$X_t = -\frac{1}{2}\beta(t)X_t t + \sqrt{\beta(t)}W_t$$

- Mean is preserved over time: $\mathbb{E}[X_t] \approx X_0$
- Variance increases gradually: noise dominates as $t \to T$
- Used in Denoising Diffusion Probabilistic Models (DDPM)

2. Variance Exploding SDE (VE-SDE)

$$X_t = \sigma(t)W_t, \quad \text{with } \sigma(t) \text{ increasing}$$

- No drift: $\mathbb{E}[X_t] = X_0$
- Variance explodes rapidly: $\mathrm{Var}(X_t) = \int_0^t \sigma^2(s)\, s$

Itô's Lemma

- Analogous to the chain rule in standard calculus.
- Let X_t satisfy: $X_t = \mu(X_t, t)t + \sigma(X_t, t)W_t$
- For a twice-differentiable function $f(X_t, t)$:

$$f(X_t, t) = \left(\frac{\partial f}{\partial t} + \mu \frac{\partial f}{\partial X} + \frac{1}{2}\sigma^2 \frac{\partial^2 f}{\partial X^2} \right) t + \sigma \frac{\partial f}{\partial X} W_t$$

- The Itô correction term $\frac{1}{2}\sigma^2 \frac{\partial^2 f}{\partial X^2}$ arises from the quadratic variation of Brownian motion.

Example: Itô's Lemma for $f(X_t) = X_t^2$

Let:

$$X_t = \mu t + \sigma W_t$$

where μ, σ are constants and W_t is standard Brownian motion.

Define: $f(X_t) = X_t^2$

Apply Itô's Lemma:

$$f(X_t) = \left(\frac{\partial f}{\partial X} \mu + \frac{1}{2} \frac{\partial^2 f}{\partial X^2} \sigma^2 \right) t + \frac{\partial f}{\partial X} \sigma W_t$$

Compute derivatives:

$$\frac{\partial f}{\partial X} = 2X_t, \quad \frac{\partial^2 f}{\partial X^2} = 2$$

Result:

$$(X_t^2) = (2X_t \mu + \sigma^2)t + 2X_t \sigma W_t$$

Insight: The σ^2 term appears only due to stochasticity — it is the **Itô correction**.

SDE for Speech Reconstruction

SDE:

$$\mathrm{d}x_t = \gamma(y - x_t)\,\mathrm{d}t + g(t)\,\mathrm{d}w \tag{6}$$

$$g(t) = \sigma_{\text{min}} \left(\frac{\sigma_{\text{max}}}{\sigma_{\text{min}}}\right)^t \sqrt{2 \log \left(\frac{\sigma_{\text{max}}}{\sigma_{\text{min}}}\right)} \tag{7}$$

- Observe this forward process takes clean speech to noisy one as $T \to \infty$.
- Noise Schedule is exponentially increasing for a fix value to another.(So, it is variance exploding *Itô* SDE)
- Also this *Itô* SDE is mean reverting.
- Important point is that for this SDE Pretubation kernel and reverse SDE can be found explicitly.

Pertubation Kernel

$$p_{0t}(x_t \mid x_0, y) = \mathcal{N}_{\mathbb{C}}\left(x_t; \mu_{x_0,y}(t), \sigma(t)^2 \mathbf{I}\right)$$

where,

$$\mu_{x_0,y}(t) = e^{-\gamma t} x_0 + \left(1 - e^{-\gamma t}\right) y$$

$$\sigma(t)^2 = \frac{\sigma_{\text{min}}^2 \left(\left(\frac{\sigma_{\text{max}}}{\sigma_{\text{min}}} \right)^{2t} - e^{-2\gamma t} \right) \log \left(\frac{\sigma_{\text{max}}}{\sigma_{\text{min}}} \right)}{\gamma + \log \left(\frac{\sigma_{\text{max}}}{\sigma_{\text{min}}} \right)}$$

Pertubation Kernel Derivation(intutive)

- First observe that noise added has zero mean at each step.
- For mean consider ODE which comes after substituting $g(t) = 0$.
- After this use initial condition of x_0 to get:

$$\mu_{x_0,y}(t) = e^{-\gamma t} x_0 + \left(1 - e^{-\gamma t}\right) y$$

- Here it could be easily seen that as $T \to \infty$, $\mu_{x_0,y}(t) \to y$.So, it is indeed mean reverting.

Pertubation Kernel Derivation(formal)

- **OU SDE**:

$$(x_t) = \gamma(y - x_t)t + g(t)d(w_t)$$

$$(e^{\gamma t}x_t) = \gamma y e^{\gamma t}t + g(t)e^{\gamma t}d(w_t)$$

$$e^{\gamma t}x_t = x_0 + \gamma y \int_0^t e^{\gamma s}s + \int_0^t g(s)e^{\gamma s}d(w_s)$$

$$x_t = x_0 e^{-\gamma t} + y(1 - e^{-\gamma t}) + \int_0^t g(s)e^{-\gamma(t-s)}d(w_s)$$

- This is of form:

$$x_t = \mu(x_0, y, t) + \epsilon_t$$

- $\mu(x_0, y, t) = x_0 e^{-\gamma t} + y(1 - e^{-\gamma t})$
- $\epsilon_t = \int_0^t g(s)e^{-\gamma(t-s)}d(w_s)$ which is a guassian with 0 mean and finite covariance.

Pertubation Kernel Derivation(formal)

- **Mean**:

$$\mathbb{E}[x_t] = x_0 e^{-\gamma t} + y(1 - e^{-\gamma t})$$

- **Covariance Matrix**:

$$\epsilon_t = \int_0^t g(s) e^{-\gamma(t-s)} d(w_s)$$

$$Var[x_t] = \mathbb{E}[x_t^2] - \mathbb{E}[x_t]^2$$

$$Var[x_t] = \mathbb{E}[\epsilon_t^2]$$

$$Var[x_t] = \int_0^t g(s)^2 e^{-2\gamma(t-s)} d(s)$$

Perturbation Kernel Derivation (Formal)

- To derive the closed-form expression for the perturbation kernel variance, we substitute the noise scale $g(t)$ into the stochastic process.
- The resulting variance of the perturbation kernel is:

$$\sigma(t)^2 = \frac{\sigma_{\text{min}}^2 \left(\left(\frac{\sigma_{\text{max}}}{\sigma_{\text{min}}} \right)^{2t} - e^{-2\gamma t} \right) \log \left(\frac{\sigma_{\text{max}}}{\sigma_{\text{min}}} \right)}{\gamma + \log \left(\frac{\sigma_{\text{max}}}{\sigma_{\text{min}}} \right)}$$

- This expression captures both exponential decay (via γ) and geometric growth (via the log-variance schedule), allowing smooth control over the noise level in diffusion-based models.

Reverse SDE

- Given forward process as:

$$\mathrm{d}x_t = f(x_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}w, \tag{8}$$

- Coresponding reverse process is given by:

$$\mathrm{d}x_t = \left[f(x_t, t) - g(t)^2 \nabla_{x_t} \log p_t(x_t)\right]\mathrm{d}t + g(t)\mathrm{d}\bar{w}, \tag{9}$$

- Interpretation:

$$x_{t+1} = x_t + f(x_t, t)\,\mathrm{d}t + g(t)\,\mathrm{d}w$$

then,

$$x_t = x_{t+1} - \left[f(x_{t+1}, t+1) - g(t+1)^2 \nabla_{x_{t+1}} \log p_{t+1}(x_{t+1})\right]\mathrm{d}t + g(t+1)\mathrm{d}$$

Fokker-Planck Equation

- The Fokker-Planck equation governs the time evolution of the probability density function $p(x, t)$ of a stochastic process.
- For an Itô SDE of the form:

$$\mathrm{d}x_t = f(x_t, t)\,\mathrm{d}t + g(x_t, t)\,\mathrm{d}w_t$$

the corresponding Fokker-Planck equation is:

$$\frac{\partial p(x,t)}{\partial t} = -\nabla_x \cdot (f(x,t)p(x,t)) + \frac{1}{2}\nabla_x^2 : (D(x,t)p(x,t))$$

- Here, the diffusion matrix is:

$$D(x,t) = g(x,t)g(x,t)^\top$$

Fokker-Planck Equation

- The Fokker-Planck equation tells us how probability "flows" in time.
- **Drift term**: $-\nabla_x \cdot (f(x,t)p(x,t))$ moves the density along vector field f.
- **Diffusion term**: $\frac{1}{2}\nabla_x^2 : (D(x,t)p(x,t))$ spreads the density (adds uncertainty).
- In score-based diffusion models, the reverse-time SDE is derived from the Fokker-Planck equation by incorporating the score function:

$$\nabla_x \log p_t(x)$$

- This forms the basis of reverse diffusion and generative sampling in SDE-based speech and image restoration models.

Reverse Process Derivation

- Some Basic Notation:
  - $P_t(x)$: Probability density function of $x(t)$.
  - **f(x,t) and g(t)**: Drift and Diffusion term respectively.
  - **T**: Total number of steps considered.
  - $\sim$: Represent corresponding terms for reverse process.
- Some important points:
  - $\tilde{x}(t) = x(T - t)$
  - $\tilde{P}_t(x) = P_{T-t}(x)$
  - $\tilde{g}(t) = g(T - t)$ (just revert the diffusion term)

Reverse Process Derivation

- Fokker plank equation for reverse Process:

$$\frac{\partial \tilde{p}_t(x)}{\partial t} = -\nabla_{\tilde{x}_t} \cdot (\tilde{f}(x,t)\tilde{p}_t(x)) + \frac{1}{2}\tilde{g}(t)\nabla_{\tilde{x}_t{}^2}\tilde{p}_t(x)$$

Substitute $s = T - t$

$$\frac{\partial p_s(x)}{\partial t} = -\nabla_{x_s} \cdot (\tilde{f}_t(x)p_s(x)) + \frac{1}{2}g(s)^2\nabla^2_{x_s}p_s(x)$$

- Fokker plank equation for forward process for s = T-t:

$$\frac{\partial p_s(x)}{\partial t} = -\frac{\partial p_s(x)}{\partial s} = \nabla_{x_s} \cdot (f_s(x)p_s(x)) - \frac{1}{2}g(s)^2\nabla^2_{x_s}p_s(x)$$

Reverse Process Derivation

- Combine above 2 equations:

$$\nabla_{x_s} \cdot (\tilde{f}_t(x) p_s(x)) = -\nabla_{x_s} \cdot (f_s(x) p_s(x)) + g(s)^2 \nabla_{x_s}^2 p_s(x)$$

- Solving this:

$$\tilde{f}_t(x) = -f_{T-t}(x) + g(T-t)^2 \nabla_{x_t} P_{T-t}(x)$$

-

Speech Enhancement Algorithm

- Observe given the noisy speech only unknown factor in reverse process is $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ which is known as score function
- If somehow we are able to learn this, then we'll be able to generate clean speech.
- Directly computing the score function $\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t)$ is intractable for complex data.
- Therefore, we train a neural network $s_\theta(\mathbf{x}_t, t)$ to approximate this score function.
- In the case of conditional Gaussian distribution $p_t(\mathbf{x}_t|\mathbf{x}_0)$, the score function is known:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_0) = -\frac{\mathbf{x}_t - \boldsymbol{\mu}(t|\mathbf{x}_0)}{\sigma^2(t)}$$

Score Function Estimation

- Suppose the noisy observation is:

$$\mathbf{x}_t = \boldsymbol{\mu}(t|\mathbf{x}_0) + \sigma(t) \cdot \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$$

- Then the score simplifies to:

$$\nabla_{\mathbf{x}_t} \log p_t(\mathbf{x}_t|\mathbf{x}_0) = -\frac{\mathbf{z}}{\sigma(t)}$$

- The neural network $s_\theta(\mathbf{x}_t, t)$ is trained to match this true score.
- The training objective becomes:

$$\mathbb{E}_{\mathbf{x}_0, \mathbf{z}, t}\left[\left\| s_\theta(\mathbf{x}_t, t) + \frac{\mathbf{z}}{\sigma(t)} \right\|^2\right]$$

- This is known as **denoising score matching**, and is a core technique in DDPMs and score-based generative models.

Significance of score function

- Langevin dynamics is a sampling method that uses the score function to generate samples from a distribution:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \frac{\eta}{2}\nabla_{\mathbf{x}} \log p(\mathbf{x}_k) + \sqrt{\eta} \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$$

- This simulates a diffusion process where particles move towards high-density regions of $p(\mathbf{x})$.
- So, score of any distribution gives us full information about corresponding distribution which is a little bit intutive as knowledge of gradient at each point and normalization constraint ensures this.
- This forms the basis for score-based generative models and denoising diffusion probabilistic models (DDPMs).

score function Animation

Forward and Reverse Processes

- Forward SDE (Ornstein-Uhlenbeck process):

$$\mathrm{d}\mathbf{x}_\tau = \gamma(\mathbf{y} - \mathbf{x}_\tau)\mathrm{d}\tau + g(\tau)\mathrm{d}\mathbf{w}$$

- Perturbation kernel solution:

$$p_{0,\tau}(\mathbf{x}_\tau|\mathbf{x}_0, \mathbf{y}) = \mathcal{N}_{\mathbb{C}}(\mathbf{x}_\tau; \boldsymbol{\mu}, \sigma(\tau)^2\mathbf{I})$$

- Closed-form mean and variance:

$$\boldsymbol{\mu} = e^{-\gamma\tau}\mathbf{x}_0 + (1 - e^{-\gamma\tau})\mathbf{y}$$

$$\sigma(\tau)^2 = \frac{\sigma_{\text{min}}^2((\sigma_{\text{max}}/\sigma_{\text{min}})^{2\tau} - e^{-2\gamma\tau})\log(\sigma_{\text{max}}/\sigma_{\text{min}})}{\gamma + \log(\sigma_{\text{max}}/\sigma_{\text{min}})}$$

Score Function Estimation

- Score function of perturbation kernel:

$$\nabla_{\mathbf{x}_\tau} \log p_{0,\tau}(\mathbf{x}_\tau|\mathbf{x}_0, \mathbf{y}) = -\frac{\mathbf{x}_\tau - \boldsymbol{\mu}}{\sigma(\tau)^2}$$

- Denoising score matching objective:

$$\mathcal{J}^{(\text{DSM})}(\phi) = \mathbb{E}\left[\left\|\mathbf{s}_\phi(\mathbf{x}_\tau, \mathbf{y}, \tau) + \frac{\mathbf{z}}{\sigma(\tau)}\right\|_2^2\right]$$

- Training process:
  - Sample $\tau \sim \mathcal{U}[\tau_\epsilon, T]$
  - Generate $\mathbf{x}_\tau = \boldsymbol{\mu} + \sigma(\tau)\mathbf{z}$
  - Train DNN $\mathbf{s}_\phi$ to estimate score

Limitations of Existing Approaches

- **Predictive Models**:
  - Dominate speech restoration, learning a deterministic mapping.
  - Can distort target speech and suffer from generalizability issues.
  - Tend to "smooth out" fine-grained details (regression to the mean).
- **Generative Models (Diffusion Models)**:
  - Show great ability in bridging performance gaps and can outperform predictive models for certain corruption types.
  - **High Computational Burden**: Require running a neural network for each reverse diffusion step.
  - Can produce vocalizing and breathing artifacts in adverse conditions.

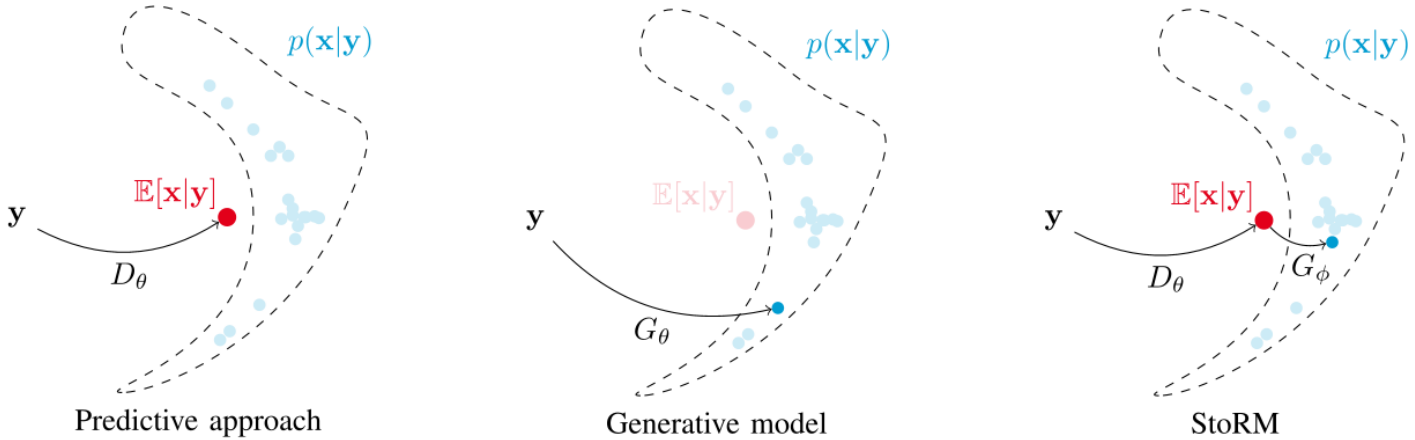
Predictive Artifacts in Speech Spectrograms

- Inverse problems (denoising, dereverberation, bandwidth extension).
- Predictive models trained with L^2 regression result in:
  - Smoothing
  - Loss of fine-grained details
  - Over-denoising
- Similar to edge loss in image denoising.

StoRM: Bridging the Gap

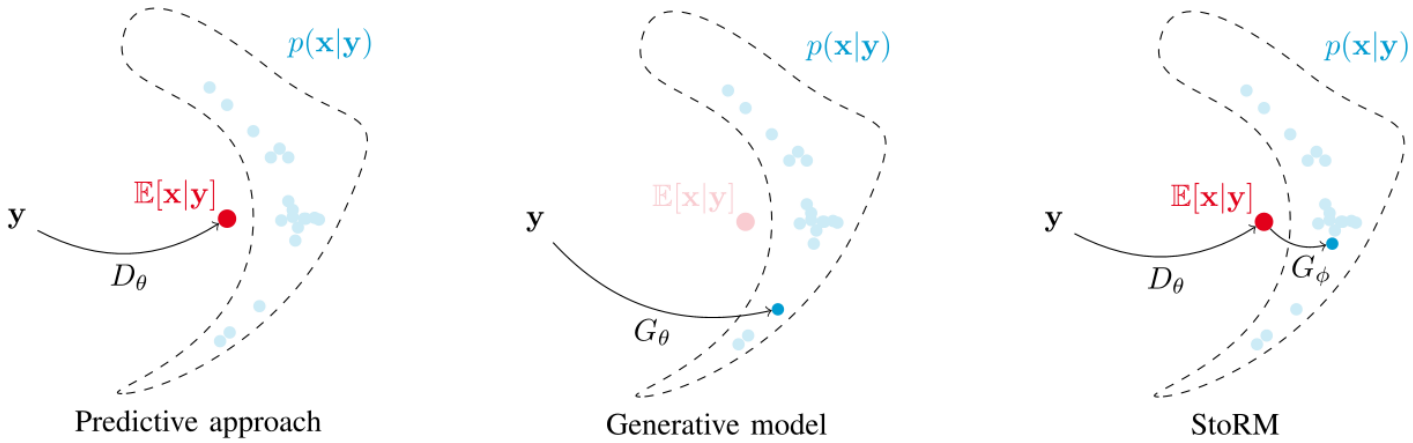
- Proposed: **Stochastic Regeneration Approach (StoRM)**.
- Combines predictive and generative models.
- Uses an estimate from a predictive model as a guide for further diffusion.
- **Advantages**:
  - Removes vocalizing and breathing artifacts.
  - Produces very high-quality samples.
  - Enables lighter sampling schemes with fewer diffusion steps, reducing computational burden by an order of magnitude.

Predictive vs. Generative Models: A Distributional View



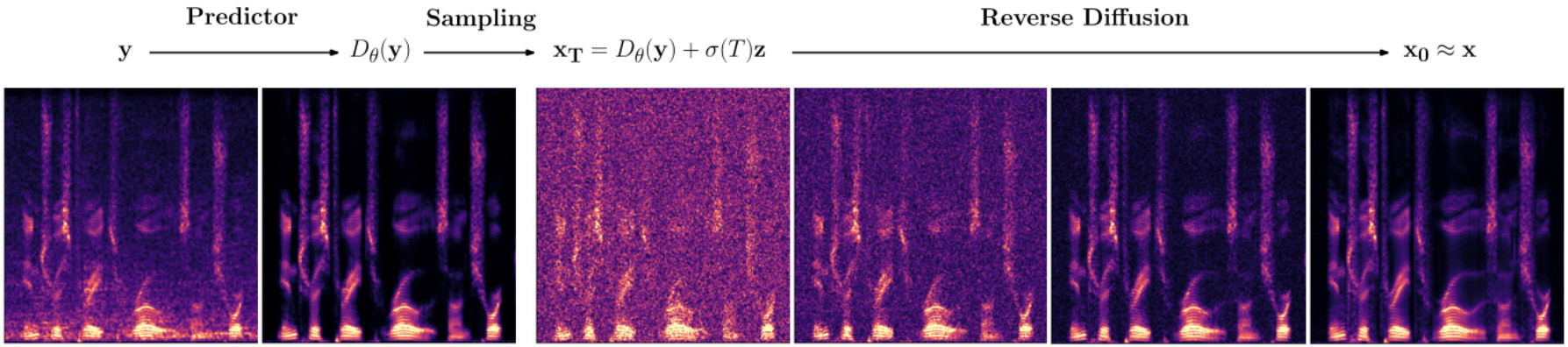
- **Left (Predictive approach)**:
  - The model D_θ predicts a single point: $\mathbb{E}[x|y]$.
  - Optimized via MSE loss; result often lies outside the high-density region of $p(x|y)$.
  - Leads to overly smooth, less natural outputs.
- **Middle (Generative model)**:
  - Attempts to sample from $p(x|y)$ using a generative model G_θ.
  - No predictive anchor $\rightarrow$ learning is harder, especially when SNR is low.
  - May require long sampling or produce unrealistic outputs if not properly regularized.

StoRM: Guided Generation for Better Modeling of $p(x|y)$



- **Right (StoRM)**:
  - Begins from $\mathbb{E}[x|y] = D_\theta(y)$ — a high-SNR estimate from the predictive model.
  - Then refines this estimate using a score-based generative model G_ϕ.
  - Final output lies in the high-density region of $p(x|y)$, capturing realistic variations.
- **Advantage:**
  - Easier learning than pure generative methods.
  - Better perceptual quality than pure predictive methods.

StoRM Inference Process Visualization



- **Left to Right**:
  - Input spectrogram y
  - Predictive estimate $D_\theta(y)$
  - Sampled noisy signal $x_T = D_\theta(y) + \sigma(T)z$
  - Intermediate reverse diffusion steps
  - Final output $x_0 \approx x$
- Demonstrates the full StoRM inference pipeline from noisy input to high-fidelity output.

Inference Pipeline in StoRM

- The StoRM inference process reconstructs clean speech x from a corrupted observation y using a two-stage approach:
  1. **Prediction Stage:** A predictive model $D_\theta(y)$ estimates an initial denoised speech signal.
  2. **Sampling Stage:** Noise is added to the estimate: $x_T = D_\theta(y) + \sigma(T)z$, where $z \sim \mathcal{N}(0, I)$.
  3. **Reverse Diffusion Stage:** A score-based generative model refines x_T via reverse-time SDE sampling to produce $x_0 \approx x$.
- This design provides:
  - High initial SNR to facilitate diffusion.
  - Reduction in artifacts like over-smoothing and vocoding effects.

Interpretation of Spectrogram Transitions

- **Leftmost Panel (y):** Corrupted speech spectrogram (e.g., noisy, reverberated input).
- **Second Panel ($D_\theta(y)$):** Initial clean speech estimate from predictive model; suffers from smoothing artifacts.
- **Middle Panel (x_T):** Sampled noisy version of the prediction — starting point for reverse diffusion.
- **Next Panels:** Intermediate results during reverse diffusion. Noise gradually removed while preserving structure.
- **Rightmost Panel (x_0):** Final output — a high-quality approximation of clean speech with better resolution and less distortion.

Diffusion Process Mathematics

Modified Forward SDE:

$$\mathrm{d}\mathbf{x}_\tau = \gamma \left(D_\theta(\mathbf{y}) - \mathbf{x}_\tau \right) \mathrm{d}\tau + g(\tau)\, \mathrm{d}\mathbf{w}$$

Perturbation Kernel:

$$p_{0,\tau}(\mathbf{x}_\tau | \mathbf{x}_0, D_\theta(\mathbf{y})) = \mathcal{N}_{\mathbb{C}}(\mathbf{x}_\tau;\, \boldsymbol{\mu}_{\text{reg}},\, \sigma(\tau)^2 \mathbf{I})$$

Where the mean is:

$$\boldsymbol{\mu}_{\text{reg}} = e^{-\gamma\tau}\, \mathbf{x}_0 + \left(1 - e^{-\gamma\tau}\right) D_\theta(\mathbf{y})$$

- This defines a Gaussian perturbation around a learned target $D_\theta(\mathbf{y})$.
- As $\tau \to \infty$, the distribution converges toward $D_\theta(\mathbf{y})$.

Score Function and Training Objective

Adapted Score Function:

$$\nabla_{\mathbf{x}_\tau} \log p_\tau(\mathbf{x}_\tau \mid D_\theta(\mathbf{y})) = -\frac{\mathbf{x}_\tau - \boldsymbol{\mu}_{\text{reg}}}{\sigma(\tau)^2}$$

Combined Training Objective (StoRM):

$$\mathcal{J}^{(\text{StoRM})} = \alpha \mathcal{J}^{(\text{Sup})} + (1-\alpha)\mathcal{J}^{(\text{DSM})}$$

Where:

$$\mathcal{J}^{(\text{Sup})} = \mathbb{E}\left[\|\mathbf{x} - D_\theta(\mathbf{y})\|_2^2\right]$$

$$\mathcal{J}^{(\text{DSM})} = \mathbb{E}\left[\left\|\mathbf{s}_\phi(\mathbf{x}_\tau, [\mathbf{y}, D_\theta(\mathbf{y})], \tau) + \frac{\mathbf{z}}{\sigma(\tau)}\right\|_2^2\right]$$

- $\mathcal{J}^{(\text{Sup})}$ encourages direct supervised reconstruction.
- $\mathcal{J}^{(\text{DSM})}$ trains the score network to match the Langevin-inspired score.
- $\alpha \in [0, 1]$ balances between supervised and denoising-score-matching losses.

Inference Procedure

1. Generate initial estimate:

$$\hat{\mathbf{x}}^{(0)} = D_\theta(\mathbf{y})$$

2. Sample starting point:

$$\mathbf{x}_T \sim \mathcal{N}_{\mathbb{C}}(\hat{\mathbf{x}}^{(0)}, \sigma^2(T)\mathbf{I})$$

3. Take reverse steps from $x_T \to x_0$ as done in simple SDE algorithm.
4. Final output: $\mathbf{x}_0$ after T steps

Pseudocode Training

Algorithm 1 StoRM Training

Input: Training set of pairs $(\mathbf{x}, \mathbf{y})$
Output: Trained parameters $\{\phi, \theta\}$

1: Sample diffusion time $t \sim \mathcal{U}(t_\epsilon, T)$
2: Sample noise signal $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
3: Infer initial prediction $D_\theta(\mathbf{y})$
4: Generate perturbed state $\mathbf{x}_\tau \leftarrow \boldsymbol{\mu}(\mathbf{x}, D_\theta(\mathbf{y}), \tau) + \sigma(\tau)\mathbf{z}$
5: Estimate score $\mathbf{s}_\phi(\mathbf{x}_\tau, [\mathbf{y}, D_\theta(\mathbf{y})], \tau)$
6: Compute loss $\mathcal{J}^{(\mathrm{StoRM})}(\phi, \theta)$ (eq. (17))
7: Backpropagate loss $\mathcal{J}^{(\mathrm{StoRM})}(\phi, \theta)$ to update $\{\phi, \theta\}$

Figure: Pseudocode(STORM TRAINING)

Pseudocode Inference

Algorithm 2 StoRM Inference (based on PC sampling by [44])

Input: Corrupted speech $\mathbf{y}$, step size $\Delta\tau = \frac{T}{N}$
Output: Clean speech estimate $\widehat{\mathbf{x}}$

Infer initial prediction $D_\theta(\mathbf{y})$
Sample noise signal $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$
Generate initial reverse state $\mathbf{x}_T = D_\theta(\mathbf{y}) + \sigma(T)\mathbf{z}$
for $n \in \{N, \ldots, 1\}$ **do**
    Get diffusion time $\tau = n\Delta\tau = \frac{n}{N}T$
    if using corrector **then**
        Estimate score $\mathbf{s}_\phi(\mathbf{x}_\tau, [\mathbf{y}, D_\theta(\mathbf{y})], \tau)$
        Sample correction noise signal $\mathbf{w}_c \sim \mathcal{N}(0, \mathbf{I})$
        Correct estimate (Annealed Langevin Dynamics):

$$\mathbf{x}_\tau \leftarrow \mathbf{x}_\tau + 2r^2\sigma(\tau)^2\mathbf{s}_\phi(\mathbf{x}_\tau, [\mathbf{y}, D_\theta(\mathbf{y})], \tau) + 2r\sigma(\tau)\mathbf{w}_c$$

Limitations of STORM for Clipped Speech

- Our objective is to handle **infinitely clipped speech**, which poses a unique challenge for traditional speech restoration models.
- The STORM model struggles in this scenario, likely because **clipping is a non-linear distortion**—unlike additive noise, it cannot be effectively modeled by simple linear perturbations.
- We hypothesize that a **Bernoulli-based diffusion model** could offer a better alternative, as it avoids the assumption of linear noise addition.
- In the following slides, we will first understand the **Bernoulli diffusion model**, and then explore potential adaptations for **clipped speech restoration**.

Bernoulli Autoencoder: Overview

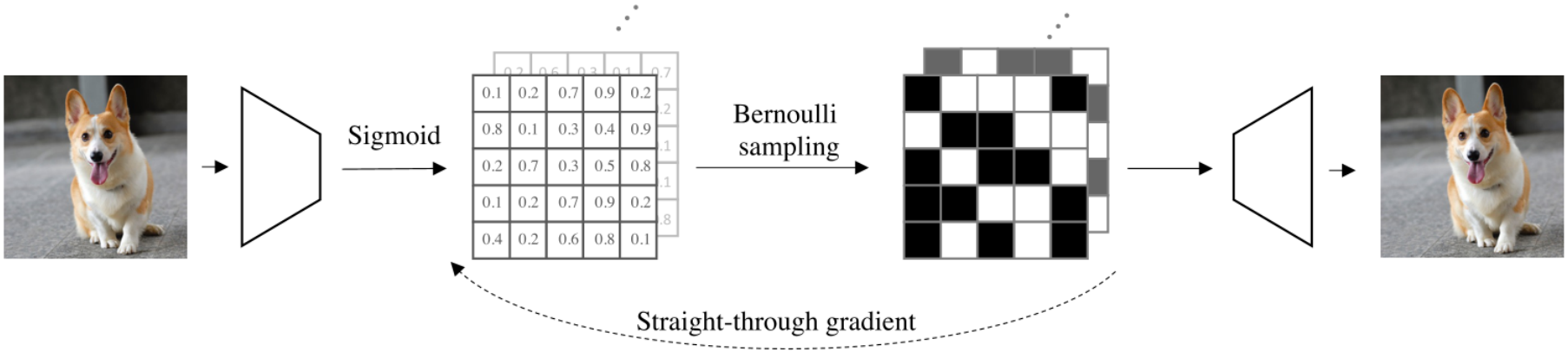


Figure: Enter Caption

- Goal: Learn a compact and expressive binary representation of the input image.
- Architecture:
  - Encoder maps input image x to a feature map $\Psi(x)$.
  - Sigmoid activation converts features to probabilities: $y = \sigma(\Psi(x))$.
  - Binary latent vector z is sampled from a Bernoulli distribution: $z \sim \text{Bernoulli}(y)$.
- Decoder reconstructs the original image from z.

Binary Representation via Bernoulli Sampling

- The encoder outputs real-valued probabilities $y \in [0, 1]$.
- Binary latent code is sampled:

$$z_i \sim \text{Bernoulli}(y_i), \quad \text{for each element } i.$$

- This introduces a discrete latent space that improves compactness and generalization.
- Sampling from Bernoulli is non-differentiable, which complicates backpropagation.

Training with Straight-Through Estimator

- To enable gradient flow through binary samples, the **Straight-Through Estimator (STE)** is used.
- Forward pass uses sampled binary z.
- Backward pass copies gradients from continuous y:

$$\tilde{z} = \text{StopGrad}(z - y) + y$$

- Allows end-to-end training despite non-differentiable sampling.

Reconstruction and Loss Objective

- Decoder reconstructs $\hat{x} = \Phi(\tilde{z})$ from the binary latent code.
- Training uses a combination of:
  - Mean Squared Error (MSE)
  - Perceptual Loss
  - Adversarial Loss (optional)
- Objective:

$$\mathcal{L} = \lambda_1 \cdot \text{MSE}(x, \hat{x}) + \lambda_2 \cdot \text{Perceptual} + \lambda_3 \cdot \text{GAN loss}$$

- Binary representations are both compact and robust for generative modeling.

Forward Diffusion Process

- **Progressive Noise Addition**: A sequence of conditional distributions $q(z^t|z^{t-1})$ progressively adds noise to the binary latent code over T steps.
- **Starting Point**: The process begins with the original binary latent code $z^0 \sim q(z^0)$, which is the output of the auto-encoder's Bernoulli sampling.
- **Bernoulli Noise Definition**:

$$q(z^t|z^{t-1}) = \mathcal{B}(z^t; z^{t-1}(1-\beta^t) + 0.5\beta^t)$$

  - β^t is a pre-defined noise schedule (e.g., increasing from a small value to 1).
  - This formulation ensures that as β^t increases, the probability of flipping bits also increases.
- **Convergence**: As $t \to T$, the latent code z^T converges to a random Bernoulli distribution:

$$z^T \sim \mathcal{B}(z^T; 0.5)$$

Reverse Diffusion Process

- **Objective**: To generate new binary latent codes, we need to reverse the diffusion process. This involves training a neural network f_θ to approximate the true reverse conditional probability $p_\theta(z^{t-1}|z^t)$.
- **Reverse Conditional Distribution**:

$$p_\theta(z^{t-1}|z^t) = \mathcal{B}(z^{t-1}; f_\theta(z^t, t))$$

  - $f_\theta(z^t, t)$ is the learned function that predicts the probability of $z^{t-1} = 1$ given z^t and the current timestep t.
- **Sampling**: By starting from a noisy sample z^T (from $\mathcal{B}(z^T; 0.5)$) and iteratively applying f_θ for T steps, we can obtain a clean binary latent code $z^0 \sim q_\theta(z^0)$.

Loss function

- It is easy to derive :

$$q(z^t \mid z^0, z^T) = \mathcal{B}(z^t; k^t z^0 + b^t), \quad \text{with}$$

$$k^t = \prod_{i=1}^{t} (1 - \beta^i),$$

$$b^t = (1 - \beta^t) b^{t-1} + 0.5\beta^t, \quad \text{and} \quad b^1 = 0.5\beta^1,$$

- It can be seen that k^t and b^t are very closely related by the following equation:

$$b^t = .5(1 - k^t)$$

- So, instead of β^t we can directly fix k^t and use it to find corresponding b^t.

Loss function

- Now, using the same derivation done for normal diffusion models:

$$\begin{aligned}\mathcal{L}_{\text{vlb}} &:= \mathcal{L}_0 + \sum_{t=1}^{T-1} \mathcal{L}_t + \mathcal{L}_T \\ &:= -\log p_\theta(z^0 \mid z^1) \\ &\quad + \sum_{t=1}^{T-1} \mathrm{KL}\left(q(z^{t-1} \mid z^t, z^0) \,\|\, p_\theta(z^{t-1} \mid z^t)\right) \\ &\quad + \mathrm{KL}\left(q(z^T \mid z^0) \,\|\, p(z^T)\right).\end{aligned}$$

- Here $q(z^{t-1} \mid z^t, z^0)$ is given by equation on previous slide.
- $p_\theta(z_{t-1} \mid z_t)$ is model output.

Reparameterization Techniques (1/2)

Predicting the Original Latent Code z^0

- **Challenge**: Direct prediction of z^{t-1} is complicated due to changing noise levels across timesteps.
- **Strategy**: Predict the original clean latent code z^0 directly.
  - The model $f_\theta(z^t, t)$ is trained to estimate $\hat{z}^0$, the original binary code.
  - This simplifies the learning target, as z^0 is a clean, noise-free binary representation.
  - Learning $P(z^0|z^t)$ is theoretically equivalent to learning $P(z^{t-1}|z^t)$.
  - However, empirical results show predicting z^0 often leads to better performance and training stability.

Reparameterization Techniques (2/2)

Predicting Residual Bit Flips

- **Strategy**: Predict the residual (bit flips) between z^t and z^0.
  - The model $f_\theta(z^t, t)$ is trained to estimate $z^t \oplus z^0$, i.e., which bits were flipped.
  - This approach leverages the sparsity of flips—especially early in the diffusion process when noise is low.
- **Advantages**:
  - Efficiently captures sparse bitwise changes.
  - Leads to more stable training and faster convergence.
  - The output of f_θ is interpreted as the probability of a bit being flipped.
- Theoretically, this approach is also equivalent to directly predicting z^{t-1} or z^0, but has practical benefits.

Reparameterization: Predicting $\mathbf{z}^0$

- Directly model $p_\theta(\mathbf{z}^0|\mathbf{z}^t)$ via:

$$\hat{\mathbf{z}}^0 = f_\theta(\mathbf{z}^t, t)$$

- Reconstruct $p_\theta(\mathbf{z}^{t-1}|\mathbf{z}^t)$ using Bayes' rule:

$$p_\theta(\mathbf{z}^{t-1}|\mathbf{z}^t) = \sum_{\mathbf{z}^0 \in \{0,1\}} q(\mathbf{z}^{t-1}|\mathbf{z}^t, \mathbf{z}^0) p_\theta(\mathbf{z}^0|\mathbf{z}^t)$$

- Explicit form:

$$p_\theta(\mathbf{z}^{t-1}|\mathbf{z}^t) = \left(\mathbf{z}^{t-1}; \frac{A}{A+B}\right)$$

where $A = [(1-\beta^t)\mathbf{z}^t + 0.5\beta^t] \odot [k^t f_\theta + 0.5 b^t]$,
$B = [(1-\beta^t)(1-\mathbf{z}^t) + 0.5\beta^t] \odot [k^t(1-f_\theta) + 0.5 b^t]$.

Residual Parameterization via XOR

- Predict flipping probability (XOR residual):

$$f_\theta(\mathbf{z}^t, t) \approx \mathbf{z}^t \oplus \mathbf{z}^0$$

- Training objective (Binary Cross-Entropy):

$$\mathcal{L}_{\text{residual}} = \mathbb{E}_{t,\mathbf{z}} \left[\text{BCE}(f_\theta(\mathbf{z}^t, t), \mathbf{z}^t \oplus \mathbf{z}^0) \right]$$

- Combined loss:

$$\mathcal{L} = \mathcal{L}_{\text{residual}} + \lambda \mathcal{L}_{\text{vlb}}$$

- λ balances diversity and stability (empirically $\lambda = 0.1$).

Pseudo-code for Bernoulli Diffusion Training

Algorithm 1 Training procedure. We assume unconditional image generation with a batch size of *one* for the sake of discussion. The described training process can be easily extended to practical cases with arbitrary batch sizes by batching multiple samples.

Given: Trained encoder $\mathbf{\Psi}$; Binary diffusion model f_θ parametrized by $\mathcal{T}_\theta$; An image dataset $\mathbf{X}$.
Given: Diffusion steps T; Noise scheduler defined by $\{k^t\}_{t=1}^T$ and $\{b_t\}_{t=1}^T$; Training steps I; and λ in (13).
Initializing $\mathcal{T}_\theta$.
for Step $i = 1 : I$ **do**
    Sampling image $\mathbf{x} \sim \mathbf{X}$, and time step $t \sim \{1, \dots, T\}$.
    Obtaining binary code $\mathbf{z}^0 = \text{Bernoulli}(\sigma(\mathbf{\Psi}(\mathbf{x})))$.
    Obtaining $\mathbf{z}^t$ using $\mathbf{z}^0$, t, and noise scheduler with (5).
    Predicting flipping probability $f_\theta(\mathbf{z}^t, t)$.
    Obtaining predicted $\mathbf{z}^0$ as $p_\theta(\mathbf{z}^0) = (1 - \mathbf{z}^t) \odot f_\theta(\mathbf{z}^t, t) + \mathbf{z}^t \odot (1 - f_\theta(\mathbf{z}^t, t))$.
    Obtaining predicted $p_\theta(\mathbf{z}^{t-1})$ using $p_\theta(\mathbf{z}^0)$ and $\mathbf{z}^t$ with (8).
    Calculating loss $\mathcal{L}$ using (13).
    Backpropagating $\mathcal{L}$ and updating θ.
end for
Return Binary diffusion model f_θ.

Figure: Pseudo-code for Bernoulli Diffusion Training

Pseudo-code for Bernoulli Diffusion Sampling

Algorithm 2 Sampling procedure. We assume unconditional image generation with a batch size of *one* for the sake o discussion.

Given: Trained decoder $\mathbf{\Phi}$; Trained binary diffusion model f_θ.
Given: Diffusion steps T; Noise scheduler defined by $\{k^t\}_{t=1}^T$ and $\{b_t\}_{t=1}^T$; Temperature τ; Latent dimension specified by h', w', e.g., $h' = w' = 16, c = 32$ for the 256×256 image generation experiments.
Sampling $\mathbf{z}^T = \text{Bernoulli}(\mathbf{z}^{\text{init}})$, where $\mathbf{z}^{\text{init}} \in \mathbb{R}^{h' \times w' \times c}$ and contains 0.5 only.
for Step $t = T : 2$ **do**
    Predicting $p_\theta(\mathbf{z}^{t-1})$ with $f_\theta(\mathbf{z}^t, t) = \sigma(\mathcal{T}_\theta(\mathbf{z}^t, t)/\tau)$ and (8).
    Sampling $\mathbf{z}^{t-1} = \text{Bernoulli}(p_\theta(\mathbf{z}^{t-1}))$
end for
Return the sampled image as $\mathbf{\Phi}(\mathbf{z}^{t-1})$.

Figure: Pseudo-code for Bernoulli Diffusion Sampling

Modification to use bernaulli diffusion model for speech restoration

- **Strategy**: We want to come up with a forward process which can take our clean speech to noisy speech in bernaulli encoded environment.
- Being able to do so ensures the reverse process to denoise our noisy speech
- Proposed Forward process:

$$q(z^t|z^{t-1}) = \mathcal{B}(z^t; z^{t-1}(1-\beta^t) + y\beta^t)$$

where y is clean version of speech.

Forward Diffusion Process(Reconstruction)

- **Progressive Noise Addition**: A sequence of conditional distributions $q(z^t|z^{t-1})$ progressively adds noise to the binary latent code over T steps.
- **Starting Point**: The process begins with the original binary latent code $z^0 \sim q(z^0)$, which is the output of the auto-encoder's Bernoulli sampling for clean speech.
- **Bernoulli Noise Definition**:

$$q(z^t|z^{t-1}) = \mathcal{B}(z^t; z^{t-1}(1-\beta^t) + y\beta^t)$$

  - β^t is a pre-defined noise schedule (e.g., increasing from a small value to 1).
  - Here y is encoding of noisy in bernoulli domain.
- **Convergence**: As $t \to T$, the latent code z^T converges to a random Bernoulli distribution:

$$z^T \sim \mathcal{B}(z^T; y)$$

Reverse Diffusion Process

- **Objective**: To generate new binary latent codes, we need to reverse the diffusion process. This involves training a neural network f_θ to approximate the true reverse conditional probability $p_\theta(z^{t-1}|z^t)$.
- **Reverse Conditional Distribution**:

$$p_\theta(z^{t-1}|z^t) = \mathcal{B}(z^{t-1}; f_\theta(z^t, t))$$

  - $f_\theta(z^t, t)$ is the learned function that predicts the probability of $z^{t-1} = 1$ given z^t and the current timestep t.
- **Sampling**: By starting from a noisy sample $z^T = y$(here y is bernaulli encoding of noisy speech) and iteratively applying f_θ for T steps, we can obtain a clean binary latent code $z^0 \sim q_\theta(z^0)$.

Loss function

- It is easy to derive :

$$q(z^t \mid z^0, z^T) = \mathcal{B}(z^t; k^t z^0 + b^t), \quad \text{with}$$

$$k^t = \prod_{i=1}^{t} (1 - \beta^i),$$

$$b^t = (1 - \beta^t) b^{t-1} + y\beta^t, \quad \text{and} \quad b^1 = y\beta^1,$$

- It can be seen that k^t and b^t are very closely related by the following equation:

$$b^t = y(1 - k^t)$$

- So, instead of β^t we can directly fix k^t and use it to find corresponding b^t.

Loss function

- Now, using the same derivation done for normal diffusion models:

$$\begin{aligned}\mathcal{L}_{\text{vlb}} &:= \mathcal{L}_0 + \sum_{t=1}^{T-1} \mathcal{L}_t + \mathcal{L}_T \\ &:= -\log p_\theta(z^0 \mid z^1) \\ &\quad + \sum_{t=1}^{T-1} \mathrm{KL}\left(q(z^{t-1} \mid z^t, z^0) \,\|\, p_\theta(z^{t-1} \mid z^t)\right) \\ &\quad + \mathrm{KL}\left(q(z^T \mid z^0) \,\|\, p(z^T)\right).\end{aligned}$$

- Here $q(z^{t-1} \mid z^t, z^0)$ is given by equation on previous slide.
- $p_\theta(z_{t-1} \mid z_t)$ is model output.

Reparameterization Techniques (1/2)

Predicting the Original Latent Code z^0

- **Challenge**: Direct prediction of z^{t-1} is complicated due to changing noise levels across timesteps.
- **Strategy**: Predict the original clean latent code z^0 directly.
  - The model $f_\theta(z^t, t)$ is trained to estimate $\hat{z}^0$, the original binary code.
  - This simplifies the learning target, as z^0 is a clean, noise-free binary representation.
  - Learning $P(z^0|z^t)$ is theoretically equivalent to learning $P(z^{t-1}|z^t)$.

Reparameterization Techniques (2/2)

Predicting Residual Bit Flips

- **Strategy**: Predict the residual (bit flips) between z^t and z^0.
  - The model $f_\theta(z^t, t)$ is trained to estimate $z^t \oplus z^0$, i.e., which bits were flipped.
  - This approach leverages the sparsity of flips—especially early in the diffusion process when noise is low.
- **Advantages**:
  - Efficiently captures sparse bitwise changes.
  - Leads to more stable training and faster convergence.
  - The output of f_θ is interpreted as the probability of a bit being flipped.
- Theoretically, this approach is also equivalent to directly predicting z^{t-1} or z^0, but has practical benefits.

Reparameterization: Predicting $\mathbf{z}^0$

- Directly model $p_\theta(\mathbf{z}^0|\mathbf{z}^t)$ via:

$$\hat{\mathbf{z}}^0 = f_\theta(\mathbf{z}^t, t)$$

- Reconstruct $p_\theta(\mathbf{z}^{t-1}|\mathbf{z}^t)$ using Bayes' rule:

$$p_\theta(\mathbf{z}^{t-1}|\mathbf{z}^t) = \sum_{\mathbf{z}^0 \in \{0,1\}} q(\mathbf{z}^{t-1}|\mathbf{z}^t, \mathbf{z}^0) p_\theta(\mathbf{z}^0|\mathbf{z}^t)$$

- Explicit form:

$$p_\theta(\mathbf{z}^{t-1}|\mathbf{z}^t) = [z^t + \beta^t(1-y)(1-2z^t)][\frac{(1-f_\theta)b^{t-1}}{z^t b^t + (1-z)(1-b^t)} + \frac{}{z^t(k^t + b}$$

Residual Parameterization via XOR

- Predict flipping probability (XOR residual):

$$f_\theta(\mathbf{z}^t, t) \approx \mathbf{z}^t \oplus x$$

- Training objective (Binary Cross-Entropy):

$$\mathcal{L}_{\text{residual}} = \mathbb{E}_{t,\mathbf{z}} \left[\text{BCE}(f_\theta(\mathbf{z}^t, t), \mathbf{z}^t \oplus x) \right]$$

- Combined loss:

$$\mathcal{L} = \mathcal{L}_{\text{residual}} + \lambda \mathcal{L}_{\text{vlb}}$$

- λ balances diversity and stability (empirically $\lambda = 0.1$).

Binary Diffusion Model - Training Procedure

Given:
  Trained encoder Ψ; Binary diffusion model f_θ parameterized by $\mathcal{T}_\theta$
  Clean dataset $\mathbf{X}$, noisy dataset $\mathbf{Y}$
  Diffusion steps T; Noise scheduler $\{k^t\}_{t=1}^T$, $\{b_t\}_{t=1}^T$
  Training steps I; regularization coefficient λ
Initialize $\mathcal{T}_\theta$
for $i = 1$ to I **do**
  Sample clean speech $\mathbf{x} \sim \mathbf{X}$ and corresponding noisy speech $\mathbf{y}$, and time step $t \sim \{1, \ldots, T\}$
  Obtain binary code $\mathbf{z}^0 \sim \text{Bernoulli}(\sigma(\Psi(\mathbf{x})))$
  Obtain $\mathbf{z}^t$ using $\mathbf{z}^0$, t, and noise scheduler
  Predict flipping probability: $f_\theta(\mathbf{z}^t, t)$
  Compute predicted $\mathbf{z}^0$ as:
    $p_\theta(\mathbf{z}^0) = (1 - \mathbf{z}^t) \odot f_\theta(\mathbf{z}^t, t) + \mathbf{z}^t \odot (1 - f_\theta(\mathbf{z}^t, t))$
  (Optional) Compute predicted $p_\theta(\mathbf{z}^{t-1})$ using above formulation
  Compute loss $\mathcal{L}$ (e.g., binary cross-entropy + regularization)
  Backpropagate $\mathcal{L}$ and update θ
end for
Return trained binary diffusion model f_θ

Binary Diffusion Model - Inference Procedure

Given:
  Trained decoder Φ; Trained binary diffusion model f_θ
  Noisy encoding $\mathbf{y}$; Diffusion steps T
  Noise scheduler $\{k^t\}_{t=1}^T$, $\{b_t\}_{t=1}^T$
  Temperature τ; Latent dimensions h', w', c

Sample $\mathbf{z}^T = \mathbf{y}$
Compute flipprob $= f_\theta(\mathbf{z}^T, T)$
Sample $\mathbf{z}^0 \sim \text{Bernoulli}(\text{flipprob} \cdot (1 - \mathbf{z}^T) + (1 - \text{flipprob}) \cdot \mathbf{z}^T)$
Return $\Phi(\mathbf{z}^0)$